WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Wednesday, December 06, 2006

Hide?	Set Name	Query	Hit Count
	DB=USPT; PLUR=YES; OP=ADJ		
☐	L13	L9 and (compare$ near native code)	0
☐	L12	L10	627
	DB=EPAB; PLUR=YES; OP=ADJ		
☐	L11	L10	2
	DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ		
☐	L10	L9 and compare$	1236
☐	L9	first virtual and second virtual	2815
☐	L8	L2 and native code	0
☐	L7	L2 and native code	0
☐	L6	first emulator and second emulator	58
	DB=USPT; PLUR=YES; OP=ADJ		
☐	L5	L2 and "native code"	0
☐	L4	L2	18
	DB=EPAB; PLUR=YES; OP=ADJ		
☐	L3	L2	1
	DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ		
☐	L2	L1 and compar$	32
☐	L1	first emulator and second emulator	58

END OF SEARCH HISTORY

Web

Results **1** - **10** of about **15** for **"first emulator" "compared" "native code"**. (**0.43** seconds)

SoftWindows 98 Review
The **first emulator** to accomplish this is Virtual PC (see article). ... And with the infusion of more PowerPC **native code** in the recent release of Mac OS 8.5 ...
os-emulation.net/softwindows98_review.html - 16k - Cached - Similar pages

Emu Programming: Re: Creating GBA Emulator
On the contrary , I wrote my **first emulator** in C++ and I ran into trouble from ... When you emit **native code** , all of this overhead can go away if you use a ...
www.retrogames.com/cgi-bin/wwwthreads/showpost.pl?Board=retroemuprog&Number=3799&page=&view=&... - 32k - Supplemental Result - Cached - Similar pages

[PDF] Study of the techniques for emulation programming
File Format: PDF/Adobe Acrobat - View as HTML
It could be said that the **first emulator** was created when the first computer ... Binary translation means to get the **native code** for the emulated CPU and ...
personals.ac.upc.edu/vmoya/docs/emuprog.pdf - Similar pages

[PDF] Study of the techniques for emulation programming
File Format: PDF/Adobe Acrobat - View as HTML
It could be said that the **first emulator** was created when the first computer ... The other way is to get the **native code** and translate it into new code for ...
people.ac.upc.edu/vmoya/docs/emuprog.pdf - Supplemental Result - Similar pages

3ddesktop src 43e1022e6d20542f965c79e23aaef22299a4b6d7 OpenGL ...
3ddesktop src 43e1022e6d20542f965c79e23aaef22299a4b6d7 OpenGL program for switching virtual desktops in 3D 3D-Desktop is an OpenGL program for switching ...
rpmfind.net/linux/dag/dries/fedora/fc2/i386/SRPMS.dries/repodata/primary.xml.gz - 250k - Supplemental Result - Cached - Similar pages

> 3ddesktop src 56fb0b128b2f9bffa86d1eb1beea63ab89d1de41 OpenGL ...
> GHC compiles Haskell to either **native code** or C. It implements numerous ... Well, everything started with Cironian who created the **first emulator** back in ...
> rpmfind.net/linux/dag/dries/fedora/fc1/i386/SRPMS.dries/repodata/primary.xml.gz - Similar pages

kernel-module-thinkpad athlon ...
kernel-module-thinkpad athlon 9004cbfe7dc40c954c4adf559ed074a4260de410 IBM ThinkPad kernel modules. IBM ThinkPad kernel modules. These drivers are built for ...
fedora.server4you.net/dag/redhat/9/en/i386/dag/repodata/primary.xml.gz - 250k - Supplemental Result - Cached - Similar pages

Ä¡Φ***%d**Φ****Θ* **Ω****∞ **φ T**±**X**≤****≈**ô**° ***² ...
File Format: Unrecognized
everything started with Cironian who created the **first emulator** back in ... retaining separate compilation), a high-performance **native code** compiler (in ...
rh-mirror.linux.iastate.edu/pub/dag/dries/fedora/fc2/i386/base/srclist.dries - Similar pages

Microsoft Word
The console shipped 22 million units **compared** with competitor PlayStation 2 at 90 million units, and the company took a 4 billion dollar loss due to the ...
2.sierpnia.pl.ogarnij.pl/en/Microsoft+Word - 250k - Supplemental Result - Cached - Similar pages

System Object Model (system object model info)
Navigation Menu. Shared Library. IBM. CORBA. Mainframe. OS/2. Microsoft Windows. Unix. AIX. VisualAge. OS/2. Workplace Shell. AIM Alliance. Apple Computer ...
system.object.model.en.xanax-prescription.be/ - 250k - Supplemental Result - Cached - Similar pages

Result Page: 1 2 **Next**

Terms used **compare native code emulator** Found **68** of **193,448**

Sort results by: relevance
Display results: expanded form
Save results to a Binder
Search Tips
Open results in a new window
Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 68 Result page: **1** 2 3 4 next

Relevance scale

1 A high performance Erlang system
Erik Johansson, Mikael Pettersson, Konstantinos Sagonas
September 2000 **Proceedings of the 2nd ACM SIGPLAN international conference on Principles and practice of declarative programming**
Publisher: ACM Press
Full text available: pdf(320.62 KB) Additional Information: full citation, references, citings, index terms

2 LiLFeS: towards a practical HPSG parser
Makino Takaki, Yoshida Minoru, Torisawa Kentaro, Tsujii Jun'ichi
August 1998 **Proceedings of the 17th international conference on Computational linguistics - Volume 2 , Proceedings of the 36th annual meeting on Association for Computational Linguistics - Volume 2**
Publisher: Association for Computational Linguistics , Association for Computational Linguistics
Full text available: pdf(521.14 KB) Publisher Site Additional Information: full citation, abstract, references, citings

This paper presents the LiLFeS system, an efficient feature-structure description language for HPSG. The core engine of LiLFeS is an Abstract Machine for Attribute-Value Logics, proposed by Carpenter and Qu. Basic design policies, the current status, and performance evaluation of the LiLFeS system are described. The paper discusses two implementations of the LiLFeS. The first one is based on an emulator of the abstract machine, while the second one uses a native-code compiler and therefore is mu ...

3 The power of partial tanslation: an experiment with the C-ification of binary Prolog
Paul Tarau, Bart Demoen, Koen De Bosschere
February 1995 **Proceedings of the 1995 ACM symposium on Applied computing**
Publisher: ACM Press
Full text available: pdf(613.38 KB) Additional Information: full citation, references, index terms

Keywords: BinWAM, WAM, compilation of binary prolog, programming language translation techniques, prolog to C translation

4 Memory compression for embedded systems: Comparing the size of .NET applications with native code
Roberto Costa, Erven Rohou
September 2005 **Proceedings of the 3rd IEEE/ACM/IFIP international conference on**

Hardware/software codesign and system synthesis CODES+ISSS '05 , Proceedings of the 3rd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis CODES+ISSS '05

Publisher: ACM Press, IEEE Computer Society

Full text available: pdf(120.62 KB) Publisher Site

Additional Information: full citation, abstract, references, index terms

Byte-code based languages are slowly becoming adopted in embedded domains because of improved security and portability. Another potential reason for their adoption is the reputation for smaller code size than native. This is critical in contexts in which a small memory footprint is crucial to reduce production costs. This paper compares the code size of applications compiled for .NET framework with the same natively compiled for various processors. The paper shows that the assumption of an impre ...

Keywords: .NET, bytecode, code size, managed environments

5 Trace-driven memory simulation: a survey

Richard A. Uhlig, Trevor N. Mudge

June 1997 **ACM Computing Surveys (CSUR)**, Volume 29 Issue 2

Publisher: ACM Press

Full text available: pdf(636.11 KB) Additional Information: full citation, abstract, references, citings, index terms, review

As the gap between processor and memory speeds continues to widen, methods for evaluating memory system designs before they are implemented in hardware are becoming increasingly important. One such method, trace-driven memory simulation, has been the subject of intense interest among researchers and has, as a result, enjoyed rapid development and substantial improvements during the past decade. This article surveys and analyzes these developments by establishing criteria for evaluating trac ...

Keywords: TLBs, caches, memory management, memory simulation, trace-driven simulation

6 Research papers III: Comparative performance analysis of mobile runtimes on Intel XScale® technology

Jason Domer, Murthi Nanja, Suresh Srinivas, Bhaktha Keshavachar

June 2004 **Proceedings of the 2004 workshop on Interpreters, virtual machines and emulators**

Publisher: ACM Press

Full text available: pdf(226.94 KB) Additional Information: full citation, abstract, references, index terms

Mobile Runtime Environments such as Java*2 Micro Edition (J2ME*) and Microsoft WinCE.NET* Compact Framework* are becoming standard managed application execution environments on memory constrained devices. A variety of implementations exists, and so too are a variety of systems they could run on, and finally a variety of workloads. It becomes important to understand how they compare.In this paper we describe comparative performance analysis of mobile runtimes on products with Intel XScale® mi ...

7 Intrusion detection: Randomized instruction set emulation to disrupt binary code injection attacks

Elena Gabriela Barrantes, David H. Ackley, Trek S. Palmer, Darko Stefanovic, Dino Dai Zovi

October 2003 **Proceedings of the 10th ACM conference on Computer and communications security**

Publisher: ACM Press

Full text available: pdf(160.71 KB) Additional Information: full citation, abstract, references, citings, index terms

Binary code injection into an executing program is a common form of attack. Most current defenses against this form of attack use a 'guard all doors' strategy, trying to block the avenues by which execution can be diverted. We describe a complementary method of protection, which disrupts foreign code execution regardless of how the code is injected. A unique and private machine instruction set for each executing program would make it difficult for an outsider to design binary attack code against ...

Keywords: automated diversity, emulation, information hiding, language randomization, obfuscation, security

8 A compiler approach to scalable concurrent-program design

Ian Foster, Stephen Taylor

May 1994 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 16 Issue 3

Publisher: ACM Press

Full text available: pdf(1.71 MB) Additional Information: full citation, abstract, references, citings, index terms, review

We describe a compilation system for the concurrent programming language Program Composition Notation (PCN). This notation provides a single-assignment programming model that permits concurrent-programming concerns such as decomposition, communication, synchronization, mapping, granularity, and load balancing to be addressed separately in a design. PCN is also extensible with programmer-defined operators, allowing common abstractions to be encapsulated and ...

Keywords: monotonicity, program composition, programming abstractions, source-to-source transformations

9 Randomized instruction set emulation

Elena Gabriela Barrantes, David H. Ackley, Stephanie Forrest, Darko Stefanović

February 2005 **ACM Transactions on Information and System Security (TISSEC)**, Volume 8 Issue 1

Publisher: ACM Press

Full text available: pdf(374.44 KB) Additional Information: full citation, abstract, references, index terms

Injecting binary code into a running program is a common form of attack. Most defenses employ a "guard the doors" approach, blocking known mechanisms of code injection. *Randomized instruction set emulation* (RISE) is a complementary method of defense, one that performs a hidden randomization of an application's machine code. If foreign binary code is injected into a program running under RISE, it will not be executable because it will not know the proper randomization. The pape ...

Keywords: Automated diversity, randomized instruction sets, software diversity

10 Mixed mode execution with context threading

Mathew Zaleski, Marc Berndl, Angela Demke Brown

October 2005 **Proceedings of the 2005 conference of the Centre for Advanced Studies on Collaborative research CASCON '05**

Publisher: IBM Press

Full text available: pdf(162.98 KB) Additional Information: full citation, abstract, references, index terms

Interpreters are widely used to implement portable language runtime environments. Programs written in these languages may benefit from performance beyond that obtainable by optimizing interpretation alone. A modern high-performance mixed-mode virtual machine (VM) includes amethod-based Just In Time (JIT) compiler. A method-based JIT, however, requires the up-front development of a complex compilation infrastructure before any performance benefits are realized.Ideally, the architecture for a mixe ...

11 The implementation of PC Scheme

David H. Bartley, John C. Jensen

August 1986 **Proceedings of the 1986 ACM conference on LISP and functional programming**

Publisher: ACM Press

Full text available: pdf(740.24 KB) Additional Information: full citation, references, citings

12 The GNU Prolog system and its implementation

Daniel Diaz, Philippe Codognet

March 2000 **Proceedings of the 2000 ACM symposium on Applied computing - Volume 2**

Publisher: ACM Press

Full text available: pdf(424.41 KB) Additional Information: full citation, references, citings, index terms

13 Finite-static code generation

Christopher W. Fraser, Todd A. Proebsting

May 1999 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1999 conference on Programming language design and implementation PLDI '99**, Volume 34 Issue 5

Publisher: ACM Press

Full text available: pdf(1.10 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper describes GBURG, which generates tiny, fast code generators based on finite-state machine pattern matching. The code generators translate postfix intermediate code into machine instructions in one pass (except, of course, for backpatching addresses). A stack-based virtual machine---known as the *Lean Virtual Machine* (LVM)---tuned for fast code generation is also described. GBURG translates the two-page LVM-to-x86 specification into a code generator that fits entirely in an 8 KB ...

14 Efficient memory management in a merged heap/stack prolog machine

Xining Li

September 2000 **Proceedings of the 2nd ACM SIGPLAN international conference on Principles and practice of declarative programming**

Publisher: ACM Press

Full text available: pdf(553.36 KB) Additional Information: full citation, references, index terms

15 Research papers III: Catenation and specialization for Tcl virtual machine performance

Benjamin Vitale, Tarek S. Abdelrahman

June 2004 **Proceedings of the 2004 workshop on Interpreters, virtual machines and emulators**

Publisher: ACM Press

Full text available: pdf(188.95 KB) Additional Information: full citation, abstract, references, index terms

We present techniques for eliminating dispatch overhead in a virtual machine interpreter using a lightweight just-in-time native-code compilation. In the context of the Tcl VM, we convert bytecodes to native Sparc code, by concatenating the native instructions used by the VM to implement each bytecode instruction. We thus eliminate the dispatch loop. Furthermore, immediate arguments of bytecode instructions are substituted into the native code using runtime specialization. Native code output fro ...

Keywords: Tcl, bytecode interpreters, just-in-time compilation, virtual machines

16 Profile-guided optimization across process boundaries

Erik Johansson, Sven-Olof Nyström
January 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN workshop on Dynamic and adaptive compilation and optimization DYNAMO '00**, Volume 35 Issue 7
Publisher: ACM Press

Full text available: pdf(911.89 KB) Additional Information: full citation, abstract, references, citings, index terms

We describe a profile-driven compiler optimization technique for *inter-process optimization*, which dynamically inlines the effects of sending messages. Profiling is used to find optimization opportunities, and to dynamically trigger recompilation and optimization at run-time. We apply the optimization technique on the concurrent programming language ERLANG, letting recompilation take place in a separate ERLANG process, and taking advantage of the facilities provided by ERLANG to dynami ...

17 Instruction merging and specialization in the SICStus Prolog virtual machine

Henrik Nässén, Mats Carlsson, Konstantinos Sagonas
September 2001 **Proceedings of the 3rd ACM SIGPLAN international conference on Principles and practice of declarative programming**
Publisher: ACM Press

Full text available: pdf(249.88 KB) Additional Information: full citation, abstract, references, citings, index terms

Wanting to improve execution speed and reduce code size of SICStus Prolog programs, we embarked on a project whose aim was to systematically investigate combination and specialization of WAM instructions. Various variants of the SICStus Prolog virtual machine instruction set were designed, implemented, and their performance was evaluated against standard benchmarks and on big Prolog programs. In this paper, we describe our methodology in finding appropriate candicates for instruction merging and ...

18 Parameter passing and control stack management in Prolog implementation revisited

Neng-Fa Zhou
November 1996 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 18 Issue 6
Publisher: ACM Press

Full text available: pdf(280.75 KB) Additional Information: full citation, abstract, references, citings, index terms

Parameter passing and control stack management are two of the crucial issues in Prolog implementation. In the Warren Abstract Machine (WAM), the most widely used abstract machine for Prolog implementation, arguments are passed through argument registers, and the information associated with procedure calls is stored in possibly two frames. Although accessing registers is faster than accessing memory, this scheme requires the argument registers to be saved and restored for back tracking and m ...

Keywords: abstract machine, prolog

19 Languages: High performance annotation-aware JVM for Java cards

Ana Azevedo, Arun Kejariwal, Alex Veidenbaum, Alexandru Nicolau
September 2005 **Proceedings of the 5th ACM international conference on Embedded software EMSOFT '05**
Publisher: ACM Press

Full text available: pdf(158.29 KB) Additional Information: full citation, abstract, references, index terms

Early applications of smart cards have focused in the area of personal security. Recently, there has been an increasing demand for networked, multi-application cards. In this new scenario, enhanced application-specific on-card Java applets and complex cryptographic services are executed through the smart card Java Virtual Machine (JVM). In order to support such computation-intensive applications, contemporary smart cards are designed with built-in microprocessors and memory. As smart cards are h ...

Keywords: Java card, high performance, superoperators, virtual machine

20 Data buffer performance for sequential Prolog architectures

E. Tick

May 1988 **ACM SIGARCH Computer Architecture News , Proceedings of the 15th Annual International Symposium on Computer architecture ISCA '88**, Volume 16 Issue 2

Publisher: IEEE Computer Society Press, ACM Press

Several local data buffers are proposed and measurements are presented for variations of the Warren Abstract Machine (WAM) architecture for Prolog. Choice point buffers, stack buffers, split-stack buffers, multiple register sets, copyback caches, and "smart" caches are examined. Statistics collected from four benchmark programs indicate that small conventional local memories perform quite well because of the WAM's high locality. The data memory performance results are equally va ...

Results 1 - 20 of 68 Result page: **1** 2 3 4 next